An Improved Approximation for Scheduling Malleable Tasks with Precedence Constraints via Iterative Method

Chi-Yeh Chen

Abstract—The problem of scheduling malleable tasks with precedence constraints is one of the most important strongly $\mathcal{NP}$-hard problems, given m identical processors and n tasks. A malleable task is one that runs in parallel on a varying number of processors. In addition, the processing sequences of tasks are constrained by the precedence constraints. The goal is to find a feasible schedule that minimizes the makespan (maximum completion time). This article presents an iterative method for improving the performance ratio of scheduling malleable tasks. The proposed algorithm achieves an approximation ratio of 4.4841 after 2 iterations. This improves the so far best-known factor of 4.7306 due to Jansen and Zhang. For a large number of iterations (> 100), the approximation ratio of the proposed algorithm is tends toward $2 + \sqrt{2} \approx 3.4143$.

Index Terms—Approximation algorithms, scheduling, malleable tasks, precedence constraints

1 Introduction

Systems with a large number of standard units are widely used for processing parallel programs. A parallel program can be represented as a set of generic malleable tasks in which each one may link to others by precedence constraints. A computational model called malleable tasks was proposed by Turek et al. [29]. A malleable task is one that runs in parallel on a varying number of processors in which processor allocation cannot change during the task execution. Many applications can be represented on malleable tasks, such as big-data analytics [3], cloud computing [4], [30], MapReduce [7], Cosmos [5], etc. The processing time of a malleable task depends on the number of allocated processors. The influence of communications, synchronization, and other overhead is taken into account in the processing time. In addition, the tasks' processing sequences are constrained by the precedence constraints. Scheduling tasks with numerous variants in precedence constraint structure (e.g., tree, series-parallel and arbitrary structure) have been studied in many papers [6], [16], [17], [21], [22].

The scheduling problem is an important issue in minimizing the makespan (maximum completion time). The problem of scheduling malleable tasks with precedence constraints is one of the most important strongly $\mathcal{NP}$-hard problems [9]. Hence, an efficient approximation algorithm with a small and provable guarantee is sought instead of an exact algorithm. The problem of scheduling malleable tasks can be considered as two problems: the allotment problem and the makespan problem. The allotment problem seeks a feasible allotment that minimizes the execution cost on a multiprocessor system. The makespan problem asks for a feasible schedule that minimizes the makespan. Many effective heuristic scheduling algorithms have been proposed for directed acyclic graph scheduling such as the Dynamic Critical Path algorithm [18], the heterogeneous earliest-finish-time (HEFT) algorithm [27], [28] and the Critical-Path-on-a-Machine (CPOP) algorithm [28]. A comprehensive review and classification of deterministic or static scheduling algorithms can be found in [19].

Du and Leung [9] demonstrated that the problem of scheduling independent malleable tasks (without precedence constraints) is strongly $\mathcal{NP}$-hard when the number of processors is 5. An approximation algorithm with a ratio of 2 for scheduling independent malleable tasks was developed by Garey and Graham [10]. Ludwig and Tiwari [23] also developed an approximation algorithm for scheduling independent malleable tasks with a ratio of 2 in an improved running time. In a series of papers, the ratio was further improved from 2 to $3/2$ by Mounié et al. [24], [25]. Jansen [13] presented an $(1.5 + \epsilon)$ approximation algorithm for scheduling malleable parallel tasks in time $O(n \log n) + f(1/\epsilon)$. Decker et al. [8] considered the problem of finding a schedule for n-independent identical malleable tasks on m identical processors and presented a 1.25-approximation algorithm for this problem. Jansen and Porkolab [14] proposed a polynomial time approximation scheme (PTAS) when the number of processors is constant. For an arbitrary number m of processors and $p(l) \leq 1, l \in \{1, \ldots, m\}$, Jansen [12] developed an asymptotic fully polynomial time approximation scheme (AFPTAS). Jansen and Thöle [15] proposed a PTAS for scheduling independent malleable tasks in which the number of processors is polynomially bounded in the

• *The author is with the Department of Computer Science and Information Engineering, National Cheng Kung University, No. 1, University Road, Tainan 701, Taiwan, ROC. E-mail: chency@csie.ncku.edu.tw.*

Manuscript received 22 Aug. 2017; revised 6 Feb. 2018; accepted 2 Mar. 2018. Date of publication 13 Mar. 2018; date of current version 8 Aug. 2018. (Corresponding author: Chi-Yeh Chen.)
Recommended for acceptance by M. Kandemir.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2018.2813387

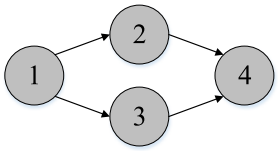

Fig. 1. A directed acyclic graph $G = (V, E)$: $V = \{1, 2, 3, 4\}$ and $E = \{(1, 2), (1, 3), (2, 4), (3, 4)\}$.

number of tasks. In addition, Barketau et al. [1] proposed polynomial time algorithms for strictly increasing convex and concave processing speed functions. They also proposed a combinatorial exponential algorithm for arbitrary strictly increasing functions in which the processing speed functions depend on the number of allocated processors.

The problem of scheduling malleable tasks with arbitrary precedence constraints is strongly $\mathcal{NP}$-hard when the number of processors is greater than or equal to 2 (Du and Leung [9]). This problem cannot be approximated within a factor of $4/3$ unless $\mathcal{P} = \mathcal{NP}$ [20]. When the work function is convex in processing time and the processing time is strictly decreasing in the number of allocated processors, Jansen and Zhang [17] devised a 3.292-approximation algorithm. Chen and Chu [6] improved the ratio of this problem to 2.9549. They also proposed an algorithm that achieves an approximation ratio of $2 + \sqrt{2} \approx 3.4143$ when the work function is convex in processing time. For tree-structured precedence constraints, the best scheduling approximation ratio was improved from 4 to 2.6181 in a series of papers [21], [22]. For general precedence constraints, a two-phase algorithm with a ratio of 5.2361 was proposed by Lepére et al. [22]. Still, Jansen and Zhang [16] devised the best-known approximation algorithm with a ratio of 4.7306.

This article develops an iterative method for improving the performance ratio of scheduling malleable tasks. A novel linear programming formulation and a rounding procedure is presented. The concept of the proposed algorithm is to reduce the gap between the fractional optimal solution and the integral optimal solution. The proposed algorithm achieves an approximation ratio of 4.4841 after 2 iterations. This improves the so far best-known factor of 4.7306 from Jansen and Zhang [16]. For a large number of iterations (> 100), the approximation ratio of the proposed algorithm tends toward $2 + \sqrt{2} \approx 3.4143$.

The rest of this article is organized as follows. Section 2 introduces basic definitions, notations, and assumptions. Section 3 reviews the previous method. Section 4 presents the proposed algorithms. Section 5 presents an approximation ratio analysis. Section 6 summarizes results concerning the approximation ratio. Section 7 draws conclusions.

2 Assumptions and Notations

A parallel program can be represented as a set of generic malleable tasks in which each task may link to the others through precedence constraints. Let $G = (V, E)$ be a directed acyclic graph (DAG), in which V is the set of malleable tasks and E is the set of precedence constraints among the tasks (see Fig. 1). For a situation in which an arc $(i, j) \in E$, task J_j cannot be processed before the finish time of task J_i. Task J_i is called a predecessor of J_j, whereas task J_j is a successor of J_i. The malleable task is one that can be executed in parallel on any integer number $l \in \{1, 2, \ldots, m\}$ of identical processors, and the corresponding discrete positive processing time is $p(l) : l \rightarrow \mathbb{R}^+$. The problem of scheduling malleable tasks with arbitrary precedence constraints requires a feasible schedule of minimum makespan (maximum completion time).

Blayo et al. [2] showed the *monotonous penalty assumptions* to be a realistic model of malleable tasks:

Assumption 1. *The processing time $p(l)$ of a malleable task is non-increasing in the number l of the allocated processors: $p(l) \leq p(l')$, for $l \geq l'$;*

Assumption 2. *The work $W(l) = w(p(l)) = l \cdot p(l)$ of a malleable task is non-decreasing in the number l of the allocated processors: $W(l) \leq W(l')$ for $l \leq l'$.*

Assumption 1 indicates that increasing the number of processors to execute a malleable task never increases the execution time. Assumption 2 indicates that increasing the number of processors to execute a malleable task leads to an increase in total overhead for communication, synchronization and scheduling.

A schedule specifies two associated values to each task J_j: the starting time τ_j and the number of allocated processors l_j. A task J_j is active during the time interval from its starting time τ_j to its completion time $C_j = \tau_j + p_j(l_j)$. A schedule is said to be feasible if, at any time $\bar{t}$, the number of active processors is at most the total number of processors $\sum_{j:\bar{t}\in[\tau_j, C_j]} l_j \leq m$ and if the precedence constraints $\tau_i + p_i(l_i) \leq \tau_j$ are satisfied for all $i \in \Gamma^-(j)$, where $\Gamma^-(j)$ is the set of predecessors of task J_j.

Table 1 presents the notation and terminology that are used herein.

3 The Previous Algorithm

Jansen and Zhang [16] devised the best-known approximation algorithm for scheduling malleable tasks with precedence constraints. The algorithm develops a linear program to solve the *allotment problem* and then runs *LIST* to schedule tasks. Based on the approximation algorithm for discrete time-cost tradeoff problems in Skutella [26], Jansen and Zhang [16] constructed virtual tasks $J_{j_1}, J_{j_2}, \ldots, J_{j_m}$ for each task J_j. Let x_{j_i} indicate the virtual processing time of virtual task J_{j_i}. For each virtual task $J_{j_i}, i = 1, \ldots, m-1$, the virtual processing time x_{j_i} is between $p_j(i)$ and 0, and the corresponding virtual work $\bar{w}_{j_i}(x_{j_i})$ is between 0 and $W_j(i+1) - W_j(i)$. The processing time of virtual task J_{j_m} is $p_j(m)$, and its virtual work $\bar{w}_{j_m}(x_{j_m})$ is zero. The following linear program is formulated to solve the allotment problem.

$$
\begin{aligned}
\min \quad & C \\
s.t. \quad & 0 \leq C_j \leq L, && \forall j; \\
& C_j + x_k \leq C_k, && \forall j \text{ and } k \in \Gamma^+(j); \\
& x_j \leq p_j(1), && \forall j; \\
& x_{j_i} \leq x_j, && \forall j \text{ and } i = 1, \cdots, m; \\
& 0 \leq x_{j_i} \leq p_j(i), && \forall j \text{ and } i = 1, \cdots, m; \\
& x_{j_m} = p_j(m), && \forall j; \\
& \bar{w}_j(x_j) = \sum_{i=1}^{m} \bar{w}_{j_i}(x_{j_i}), && \forall j; \\
& P = \sum_{j=1}^{n} p_j(1); \\
& \sum_{j=1}^{n} \bar{w}_j(x_j) + P \leq W; \\
& L \leq C; \\
& W/m \leq C;
\end{aligned}
\tag{1}
$$

TABLE 1
Notation and Terminology

n, m	The number of tasks and the number of processors.
$p_j(l)$	The processing time of a malleable task J_j in the number l of the allocated processors.
$W_j(l)$	The work of a malleable task J_j in the number l of the allocated processors.
$w_j(x_j)$	The work of a malleable task J_j in the processing time x_j.
$\Gamma^-(j), \Gamma^+(j)$	The set of predecessors of task J_j and the set of direct successors of task J_j.
x_j	The processing time of task J_j.
x_{j_i}	The virtual processing time of virtual task J_{j_i}.
$x^u_{j_i}, w^u_{j_i}$	The upper bound of processing time and the upper bound of work for the virtual tasks J_{j_i}.
$\bar{w}_{j_i}(x_{j_i})$	The virtual work of virtual task J_{j_i} in the virtual processing time x_{j_i} where $\bar{w}_{j_i}(x_{j_i}) = (W_j(i+1) - W_j(i))(1 - \frac{x_{j_i}}{p_j(i)})$ in (1) and $\bar{w}_{j_i}(x_{j_i}) = w^u_{j_i}(1 - \frac{x_{j_i}}{p_j(i)})$ in (2).
$\bar{w}_j(x_j)$	$\bar{w}_j(x_j) = \sum_{i=1}^{m} \bar{w}_{j_i}(x_{j_i})$ is the reduced work of task J_j in the processing time x_j.
$\ddot{W}, \ddot{w}_j$	The total work and the work of task J_j that is from the linear program solution in the previous iteration.
x^*_j	The processing time of task J_j that is from the linear program.
$\hat{x}_j$	The processing time of task J_j that is from the work of task J_j in the linear program.
ρ, ρ_k	The rounding parameter and the rounding parameter at the iteration k.
μ	The allotment parameter.
t	The number of iterations.
α', α	The allotment in the first phase and the allotment in the second phase.
L^*_k, W^*_k	The optimal critical path lengths and the optimal total work obtained by the linear program at the iteration k.
$L^*, W^*, C^*_{\max}$	$L^* = \max_k\{L^*_k\}, W^* = \max_k\{W^*_k\}$ and $C^*_{\max} = \max\{L^*, W^*/m\}$.
$OPT, C_{\max}$	The overall integral optimal makespan and the makespan of the final schedule.
L', W'	The critical path lengths and the total work that are obtained by the first phase.
L, W	The critical path lengths and the total work that are obtained by using the proposed algorithm.
P	$P = \sum_{j=1}^{n} p_j(1)$ is the sum of processing times of all tasks that are executed on a processor.

where $\bar{w}_j(\cdot)$ represents the *reduced work*, and $\bar{w}_{j_i}(\cdot)$ is the *virtual work* function for the fractional solution:

$$\bar{w}_{j_i}(x_{j_i}) = (W_j(i+1) - W_j(i))\left(1 - \frac{x_{j_i}}{p_j(i)}\right)$$

for all j.

However, the linear program (1) underestimates the fractional work. For example, one solution is $p_j(i+2) \le x_j \le p_j(i+1)$ which means that the program allotted more than $i+1$ processors to the task J_j. The virtual work of virtual task J_{j_i} should be $W_j(i+1) - W_j(i)$. But, since x_{j_i} may not be zero, $\bar{w}_{j_i}(x_{j_i}) = (W_j(i+1) - W_j(i))(1 - \frac{x_{j_i}}{p_j(i)}) \le W_j(i+1) - W_j(i)$ in the linear program (1). The virtual work of virtual task J_{j_i} is underestimated. Hence, this article proposes an iterative method to reduce the gap between the fractional optimal solution and the integral optimal solution.

4 Approximation Algorithm

This section develops a two-phase approximation algorithm. The first phase uses a linear program to solve the allotment problem. Let $\alpha' : V \to [1, m]$ be an allotment in the first phase that decides how many processors are allotted to execute the tasks. By rounding the fractional solution of the linear program with the parameter $\rho \in [0, 0.5]$, a feasible allotment α' can be obtained such that each task J_j is allotted the number l'_j of processors. The rounding procedure is described in detail in page 5. Let L' and W' denote the critical path lengths and the total work obtained by the first phase, respectively. The first phase runs Algorithm $\mathbb{A}$ (Allotment) to provide a feasible allotment for a given instance.

Algorithm. $\mathbb{A}$ (n, m)

1: **for** $j \leftarrow 1$ to n **do**
2: **for** $i \leftarrow 1$ to $m-1$ **do**
3: $x^u_{j_i} \leftarrow p_j(i)$;
4: $w^u_{j_i} \leftarrow W_j(i+1) - W_j(i)$;
5: $x^u_{j_m} \leftarrow p_j(m)$ and $W' \leftarrow \infty$;
6: **for** $j \leftarrow 1$ to n **do**
7: $\ddot{w}_j \leftarrow p_j(1)$;
8: $\ddot{W} \leftarrow \sum_{j=1}^{n} p_j(1)$
9: **for** $k \leftarrow 1$ to t **do**
10: Compute the linear program (2);
11: Use the rounding procedure with the rounding parameter ρ_k to obtain an allotment $\tilde{\alpha}$ and the corresponding total work $\tilde{W}$;
12: **if** $\tilde{W} < W'$ **then**
13: $W' \leftarrow \tilde{W}$ and $\alpha' \leftarrow \tilde{\alpha}$;
14: **for** $j \leftarrow 1$ to n **do**
15: $\bar{w}_j(x^*_j) = \sum_{i=1}^{m} \bar{w}_{j_i}(x^*_{j_i})$;
16: Find a processing time $\hat{x}_j$ such that $w_j(\hat{x}_j) = \bar{w}_j(x^*_j) + \ddot{w}_j$;
17: **for** $i \leftarrow 1$ to $m-1$ **do**
18: **if** $p_j(i) > \hat{x}_j$ **then**
19: **if** $p_j(i+1) < \hat{x}_j$ **then**
20: $x^u_{j_i} \leftarrow \hat{x}_j$;
21: $w^u_{j_i} \leftarrow W_j(i+1) - \bar{w}_j(x^*_j) - \ddot{w}_j$;
22: **else**
23: $x^u_{j_i} \leftarrow 0$;
24: $w^u_{j_i} \leftarrow 0$;
25: **for** $j \leftarrow 1$ to n **do**
26: $\ddot{w}_j \leftarrow \bar{w}_j(x^*_j) + \ddot{w}_j$;
27: $\ddot{W} \leftarrow \sum_{j=1}^{n} \ddot{w}_j$
28: **return** the allotment α'.

Let $\ddot{W}$ and $\ddot{w}_j$ be the total work of the linear program solution and the work of the linear program solution for the task J_j, in which the linear program solution is from the previous iteration. Based on the approximation algorithm for the discrete time-cost tradeoff problem in Skutella [26] and Jansen and Zhang's [16] idea, this article constructs virtual tasks $J_{j_1}, J_{j_2}, \ldots, J_{j_m}$ for each task J_j. The linear program (2) is a formulation of the allotment problem. Let $x^u_{j_i}$ be the upper bound of processing time and $w^u_{j_i}$ be the upper bound of work for the virtual tasks J_{j_i}. Recall that x_{j_i} indicates the virtual processing time of virtual task J_{j_i}. For each virtual task $J_{j_i}, i = 1, \ldots, m-1$, the virtual processing time x_{j_i} is

between $x^u_{j_i}$ and 0, and the corresponding virtual work $\bar{w}_{j_i}(x_{j_i})$ is between 0 and $w^u_{j_i}$. The processing time of virtual task J_{j_m} is $x^u_{j_m} = p_j(m)$, and its virtual work $\bar{w}_{j_m}(x_{j_m})$ is zero. The goal is to find an allotment that minimizes the cost $c(\alpha') = \max\{L', W'/m\}$. Therefore, the following linear program must be solved.

$$
\begin{array}{lll}
\min & C & \\
s.t. & 0 \le C_j \le L, & \forall j; \\
& C_j + x_k \le C_k, & \forall j \text{ and } k \in \Gamma^+(j); \\
& x_j \le p_j(1), & \forall j; \\
& x_{j_i} \le x_j, & \forall j \text{ and } i = 1, \cdots, m; \\
& 0 \le x_{j_i} \le x^u_{j_i}, & \forall j \text{ and } i = 1, \cdots, m; \\
& x_{j_m} = x^u_{j_m}, & \forall j; \\
& \bar{w}_j(x_j) = \sum_{i=1}^{m} \bar{w}_{j_i}(x_{j_i}), & \forall j; \\
& \sum_{j=1}^{n} \bar{w}_j(x_j) + \ddot{W} \le W; & \\
& L \le C; & \\
& W/m \le C, &
\end{array}
\tag{2}
$$

where $\bar{w}_j(\cdot)$ represents the *reduced work*, and $\bar{w}_{j_i}(\cdot)$ is the *virtual work* function for the fractional solution:

$$
\bar{w}_{j_i}(x_{j_i}) = w^u_{j_i}\left(1 - \frac{x_{j_i}}{p_j(i)}\right) \tag{3}
$$

for all j.

According to the slope formula, the following steps define a piecewise work function $w_j(x)$ in fractional processing time x_j for task J_j. If $x_j = p_j(k)$ for some $k \in \{1, \ldots, m\}$, then $w_j(x_j) = lp_j(l)$, in which $l = \min\{k | k \in \{1, \ldots, m\}, p_j(k) = x_j\}$. Otherwise, we find $l \in \{1, \ldots, m-\delta\}$ and $\delta \in \{1, \ldots, m-1\}$ such that $p_j(l-1) > p_j(l) = \cdots = p_j(l+\delta-1) > x_j > p_j(l+\delta)$ for $l > 1$ or $p_j(l) = \ldots = p_j(l+\delta-1) > x_j > p_j(l+\delta)$ for $l = 1$. We have

$$
w_j(x_j) = \frac{\delta p_j(l) p_j(l+\delta)}{p_j(l) - p_j(l+\delta)} - \frac{(l+\delta)p_j(l+\delta) - lp_j(l)}{p_j(l) - p_j(l+\delta)} x_j.
$$

In the initialization, $x^u_{j_i}$ is $p_j(i)$ and $w^u_{j_i}$ is $W_j(i+1) - W_j(i)$ for all $j \in \{1, \ldots, n\}$ and $i \in \{1, \ldots, m-1\}$. The value of $x^u_{j_m}$ is $p_j(m)$. The initial value of $\ddot{W}$ is $\sum_{j=1}^{n} p_j(1)$, and the initial value of $\ddot{w}_j$ is $p_j(1)$. Hence, in the first iteration, the linear program (2) is the same as that of Jansen and Zhang [16].

Since the linear program (1) underestimates the fractional work, the concept of Algorithm $\mathbb{A}$ is to reduce the gap between the fractional optimal solution and the integral optimal solution. Since the optimal total work is at least $\ddot{W}$, Algorithm $\mathbb{A}$ sets $\ddot{W}$ as the lower bound of total work. Let t be the number of iterations. Algorithm $\mathbb{A}$ finds a minimal total work rounding solution (lines: 12-13).

Let x^*_j be the solution of linear program (2). Let $\hat{x}_j$ be the processing time in which $w_j(\hat{x}_j) = \bar{w}_j(x^*_j) + \ddot{w}_j$. The processing time $\hat{x}_j$ can be obtained by the following steps. For any task J_j, we set $l = m$ if $\bar{w}_j(x^*_j) + \ddot{w}_j = W_j(m)$. Otherwise, we find $l \in \{1, \ldots, m-1\}$ and $\delta \in \{1, \ldots, m-1\}$ such that $W_j(l) \le \bar{w}_j(x^*_j) + \ddot{w}_j < W_j(l+\delta)$ where $p_j(l-1) > p_j(l) = \ldots = p_j(l+\delta-1) > p_j(l+\delta)$ for $l > 1$ or $p_j(l) = \ldots = p_j(l+\delta-1) > p_j(l+\delta)$ for $l = 1$. We have

$$
\hat{x}_j = \begin{cases} p_j(m), & \text{if } l = m; \\ p_j(l) + \frac{\bar{w}_j(x^*_j) + \ddot{w}_j - W_j(l)}{W_j(l+\delta) - W_j(l)}(p_j(l+\delta) - p_j(l)), & \text{if } l \ne m. \end{cases}
$$

After computing the linear program (2), we update $x^u_{j_i}$ and $w^u_{j_i}$ (lines: 14-24). We find a processing time $\hat{x}_j$ such that $w_j(\hat{x}_j) = \bar{w}_j(x^*_j) + \ddot{w}_j$. For each $i \in \{1, \ldots, m-1\}$ and $\delta \in \{0, \ldots, m-2\}$, if $p_j(i+1) < \hat{x}_j < p_j(i) = \cdots = p_j(i-\delta) < p_j(i-\delta-1)$ for $i - \delta > 1$ or $p_j(i+1) < \hat{x}_j < p_j(i) = \cdots = p_j(i-\delta)$ for $i - \delta = 1$, then the lower bound of work for task J_j is in the range $[W(i-\delta), W_j(i+1)]$ and the number of allocated processors for the task J_j is at least i. We set $x^u_{j_i} = \hat{x}_j$, $w^u_{j_i} = W_j(i+1) - \bar{w}_j(x^*_j) - \ddot{w}_j$. We also set $x^u_{j_{i'}} = 0$ and $w^u_{j_{i'}} = 0$ for all $i' < i$. Finally, we set a new lower bound of total work (lines: 25-27).

Let $\rho \in [0, 0.5]$ be the rounding parameter and ρ_k be the rounding parameter for the fractional solution at the iteration k. The rounded solution can be obtained by the following rounding procedure. For any solution $x^*_{j_i} \in [0, x^u_{j_i}]$,

$$
x_{j_i} = \begin{cases} 0, & \text{if } x^u_{j_i} = 0; \\ 0, & \text{if } x^*_{j_i} < \rho_k \cdot x^u_{j_i}; \\ p_j(i), & \text{otherwise.} \end{cases}
$$

The rounded solution is $x_j = \max_{1 \le i \le m} x_{j_i} \in \{p_j(m), \ldots, p_j(1)\}$, which can identify the allotted processors l'_j such that $p_j(l'_j) = x_j$. In the iteration k, allotment $\tilde{\alpha}$ and the corresponding total work $\tilde{W}$ can be obtained by the rounding procedure with the rounding parameter ρ_k. Thus, t rounding solutions can be obtained. The allotment α' is a minimal total work rounding solution in the t rounding solutions. We set $\rho_1 = \rho$ and $\rho_k = 2\rho$ for $k = 2, \ldots, t$.

The second phase first generates a novel allotment α, which is based on the resulting allotment α' and the allotment parameter $\mu \in [1, (m+1)/2]$. Each task J_j is allotted a number, $l_j = \min\{l'_j, \mu\}$, of processors. The second phase then runs *LIST* (as developed in Graham [11], Lepére et al. [22], and Jansen and Zhang [16]) and a feasible scheduling is provided for a given instance. Let $L, W, C_{\max}$ denote the critical path lengths, the total work, and the makespan of the final schedule obtained by the proposed algorithm.

Algorithm. LIST (J, m, α', μ)

1: Initialization: allot $l_j = \min\{l'_j, \mu\}$ processors to task J_j, for $j \in \{1, \cdots, n\}$;
2: SCHEDULED = $\emptyset$;
3: **for** SCHEDULED $\ne J$ **do**
4: READY=$\{J_j | \Gamma^-(j) \subseteq$ SCHEDULED$\}$;
5: compute the earliest possible starting time under α for all tasks in READY;
6: schedule the task $J_j \in$READY with the smallest earliest starting time;
7: SCHEDULED = SCHEDULED $\cup \{J_j\}$;

Example. Assume we have a DAG in Fig. 1. The processing times of task J_j are $p_j(1) = 34$, $p_j(2) = 20$, $p_j(3) = 14$, $p_j(4) = 12$, and $p_j(5) = 10$ for $j = 1, 2, 3, 4$. The works of task J_j are $W_j(1) = 34$, $W_j(2) = 40$, $W_j(3) = 42$, $W_j(4) = 48$, and $W_j(5) = 50$ for $j = 1, 2, 3, 4$. In the initialization of Algorithm $\mathbb{A}$, $x^u_{j_i}$ is set to $p_j(i)$ and $w^u_{j_i}$ is set to $W_j(i+1) - W_j(i)$ for all $j \in \{1, \ldots, 4\}$ and $i \in \{1, \ldots, 4\}$ (lines: 1-4). The value of $x^u_{j_5}$ is $p_j(5)$ and the value of W' is infinity. The initial value of $\ddot{W}$ is $\sum_{j=1}^{n} p_j(1)$, and the initial value of $\ddot{w}_j$ is $p_j(1)$ (lines: 6-8 in Algorithm $\mathbb{A}$).

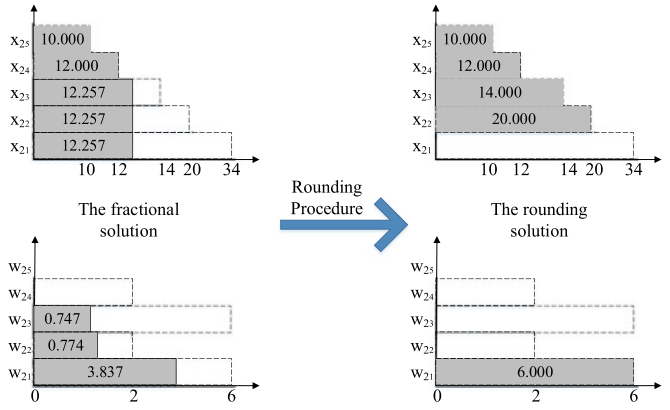

Fig. 2. The solution of task J_2 in $k = 1$; the upper bounds are represented in dashed lines.

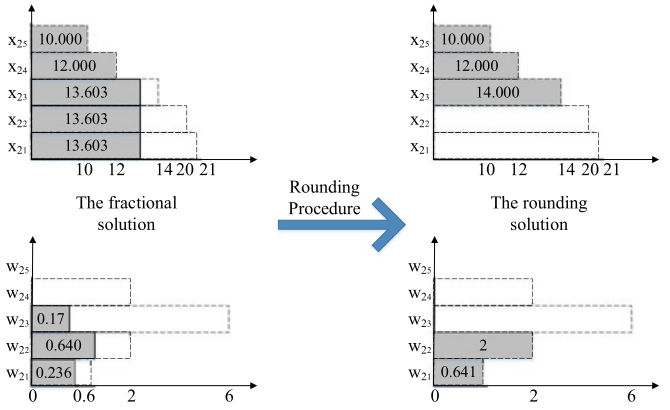

Fig. 3. The solution of task J_2 in $k = 2$; the upper bounds are represented in dashed lines.

Algorithm $\mathbb{A}$ iteratively solves the linear program (2), and updates the parameters of tasks (lines: 9-27). By computing the linear program (2), a fractional solution can be obtained. Fig. 2 shows the solution of task J_2 in $k = 1$. By using the rounding procedure with $\rho_1 = \rho = 0.4$, $x_{2_1} = 0$ since $x^*_{2_1} = 12.257 < \rho_1 \cdot x^u_{2_1} = 13.6$. Also, we can obtain $x_{2_2} = p_2(2)$, $x_{2_3} = p_2(3)$, $x_{2_4} = p_2(4)$, and $x_{2_5} = p_2(5)$. The rounding solution for the processing time is $x_2 = \max_{1\le i\le 5} x_{2_i} = 20$. The rounding solution of other tasks also can be obtained. Thus, the rounding solution for total work is $\tilde{W} = 160$ and $\tilde{\alpha} = \{l'_1, l'_2, l'_3, l'_4\} = \{2, 2, 2, 2\}$. Since $\tilde{W} < W'$, we have $W' = \tilde{W}$ and $\alpha' = \tilde{\alpha}$ (lines: 10-13 in Algorithm $\mathbb{A}$).

Algorithm $\mathbb{A}$ finds the value of $\hat{x}_j$ for $j = 1, 2, 3, 4$ and updates the parameters of tasks (lines: 14-24). Since the parameters of tasks are all updated in the same way, we only show the updating steps of task J_2 as example. The work of task J_2 in the linear program is $\bar{w}_2(x^*_2) + \ddot{w}_2 = 5.359 + 34 = 39.359$. Since $W_2(1) \le 39.359 < W_2(2)$, $\hat{x}_2 = p_2(1) + \frac{39.359 - W_2(1)}{W_2(2) - W_2(1)}(p_2(2) - p_2(1)) = 21.497$. Since $p_2(2) < \hat{x}_2 < p_2(1)$, we update $x^u_{2_1} = 21.497$ and $w^u_{2_1} = W_2(2) - \bar{w}_2(x^*_2) - \ddot{w}_2 = 40 - 39.359 = 0.641$. In the end of the first iteration, we set $\ddot{w}_1 = 41.283$, $\ddot{w}_2 = \bar{w}_2(x^*_2) + \ddot{w}_2 = 39.359$, $\ddot{w}_3 = 39.359$, $\ddot{w}_4 = 41.283$ and $\ddot{W} = \sum_{j=1}^{4} \ddot{w}_j = 161.283$ (lines: 25-27 in Algorithm $\mathbb{A}$).

We then run the second iteration. Since the parameters of tasks have been updated in the first iteration, a new fractional solution can be obtained by computing the linear program (2). Fig. 3 shows the solution of task J_2 in $k = 2$. By using the rounding procedure with $\rho_2 = 2 \cdot \rho = 0.8$, $x_{2_1} = 0$ since $x^*_{2_1} = 13.603 < \rho_2 \cdot x^u_{2_1} = 17.2$. We also can obtain $x_{2_2} = 0$, $x_{2_3} = p_2(3)$, $x_{2_4} = p_2(4)$ and $x_{2_5} = p_2(5)$. The rounding solution for the processing time is $x_2 = 14$. The rounding solution of other tasks also can be obtained. Thus, the rounding solution for total work is $\tilde{W} = 180$ and $\tilde{\alpha} = \{4, 3, 3, 4\}$. Algorithm $\mathbb{A}$ finds a minimal total work rounding solution (lines: 12-13), which is the solution in the first iteration.

5 Analysis of the Approximation Algorithm

This section analyzes the approximation algorithm. The method of analysis follows Jansen and Zhang [16]. The following steps analyze the complexity of the proposed algorithm. The first phase runs t iterations. In each iteration, the linear program (2) has $O(mn)$ variables and $O(n^2 + mn)$ constraints, and the rounding step requires, at most, $O(mn)$ time. The updating $x^u_{j_i}$ and $w^u_{j_i}$ require, at most, $O(mn)$ time. In the second phase, algorithm *LIST* has a complexity of $O(mn)$. Thus, the total runtime is $O(tmn + t \cdot LP(mn, n^2 + mn))$, where $LP(p, q)$ is the time to solve a linear program with p variables and q constraints. The major space complexity is the size of the matrix in the linear program (2), which is the number of variables times the number of constraints. Thus, the space complexity of the proposed algorithm is $O(mn^3 + m^2n^2)$. Both the time complexity and the space complexity are polynomial in the number of tasks and the number of processors.

Let L^*_k and W^*_k be optimal critical path lengths and the optimal total work obtained by the linear program (2) at iteration k. Let $L^* = \max_k\{L^*_k\}$, $W^* = \max_k\{W^*_k\}$ and $C^*_{\max} = \max\{L^*, W^*/m\}$. Let OPT be the overall integral optimal makespan. Let w^*_j be the work of task J_j obtained by the linear program (2). Since the work of task in the linear program (2) is underestimated, there exists a fractional optimal solution between the solution of linear program (2) and the integral optimal solution. For each task J_j, the work of task J_j is at least w^*_j in the fractional optimal solution, otherwise it results a contradiction that the solution of linear program (2) is optimal. Then, the work of task J_j in the fractional optimal solution is between w^*_j and $W_j(m)$, and the processing time of task J_j in the fractional optimal solution is between $p_j(m)$ and $\hat{x}_j$. The solution of linear program (2) in the next iteration is less than OPT. Therefore, OPT should be bound by the fractional optimal solution:

$$\max\{L^*, W^*/m\} \le C^*_{\max} \le OPT. \tag{4}$$

In the case of $W^*/m \ge L^*$, the processing time of virtual task J_{j_i} is equal to $\min\{x^*_j, x^u_{j_i}\}$ for obtaining the minimum total work. In the case of $W^*/m < L^*$, the processing time of virtual task J_{j_i} may not be equal to $\min\{x^*_j, x^u_{j_i}\}$. However, the solution can be adjusted to $x^*_{j_i} = \min\{x^*_j, x^u_{j_i}\}$, in which the solution $C^*_{\max} = \max\{L^*, W^*/m\}$ is not changed. Hence, we can assume that $x^*_{j_i} = \min\{x^*_j, x^u_{j_i}\}$ for all virtual tasks. The following property therefore can be obtained.

Lemma 5.1. *If $x^*_{j_i} \ge \beta \cdot x^u_{j_i}$ for some $\beta \in [0, 1]$, then $x^*_{j_k} \ge \beta \cdot x^u_{j_k}$, for any $k > i$.*

Proof. For any $k > i$ and some $\beta \in [0, 1]$, if $x^*_{j_i} \ge \beta \cdot x^u_{j_i}$ and $x^*_{j_k} < \beta \cdot x^u_{j_k}$, the work of $x^*_{j_k}$ is not less than another

solution $\bar{x}^*_{j_k} = \beta \cdot x^u_{j_k}$. Since $x^*_{j_i} \geq \beta \cdot x^u_{j_i} \geq \beta \cdot x^u_{j_k}$, $\bar{x}^*_{j_k} \leq x^*_{j_i} \leq x^*_j$ can be obtained. The solution $\bar{x}^*_{j_k}$ has less work than $x^*_{j_k}$, in which the critical path length is not changed. This results in a contradiction where $x^*_{j_k}$ is optimal. □

Let $P = \sum_{j=1}^{n} p_j(1)$ be the sum of processing times of all tasks that are executed on a processor. The bounds on L' and W' are as follows:

Lemma 5.2. *$L' \leq \frac{L^*}{\rho}$ and $W' \leq \frac{W^*}{1-2^{t-1}\rho^t} - \frac{2^{t-1}\rho^t}{1-2^{t-1}\rho^t} P$.*

Proof. Based on the rounding procedure, if $x^*_{j_i} < \rho_k \cdot x^u_{j_i}$, then $x_{j_i} = 0$, which implies that its processing time is not increasing while the work may increase. On the other hand, if $x^*_{j_i} \geq \rho_k \cdot x^u_{j_i}$, then $x_{j_i} = p_j(i)$. In the first iteration $k = 1$, $x^*_{j_i} \geq \rho_k \cdot x^u_{j_i} = \rho_k \cdot p_j(i) = \rho_k \cdot x_{j_i}$. The critical path length increases by a factor of $\frac{1}{\rho_k}$ at most by rounding. In the other iteration, $k \neq 1$, $x^*_{j_i} \geq \rho_k \cdot x^u_{j_i} \geq \rho_k \cdot p_j(i+1) \geq \rho_k \cdot \frac{i}{i+1} \cdot p_j(i) \geq \rho_k \cdot \frac{1}{2} \cdot p_j(i)$, where $p_j(i+1) \geq \frac{i}{i+1} \cdot p_j(i)$ is according to Assumption 2. The critical path length increases with a factor of $\frac{2}{\rho_k}$ at most by rounding. We have $L' \leq \frac{L^*}{\rho_k}$ for $k = 1$ and $L' \leq \frac{2L^*}{\rho_k}$ for $k \neq 1$. Since $\rho_1 = \rho$ and $\rho_k = 2\rho$ for $k \neq 1$, $L' \leq \frac{L^*}{\rho}$ can be obtained.

We now prove the upper bound of W'. If $x^*_{j_i} \geq \rho_k \cdot x^u_{j_i}$, then $x_{j_i} = p_j(i)$. Since $p_j(i) \geq x^u_{j_i}$, $\bar{w}_{j_i}(x_{j_i}) = w^u_{j_i}(1 - \frac{p_j(i)}{x^u_{j_i}}) \leq 0$. If $x^*_{j_i} < \rho_k \cdot x^u_{j_i}$, then $x_{j_i} = 0$ and $\bar{w}_{j_i}(x_{j_i}) = w^u_{j_i}$. Thus, the following chain of inequalities can be obtained:

$$\begin{aligned} \bar{w}_{j_i}(x^*_{j_i}) &= w^u_{j_i}\left(1 - \frac{x^*_{j_i}}{x^u_{j_i}}\right) \\ &\geq w^u_{j_i}\left(1 - \frac{\rho_k \cdot x^u_{j_i}}{x^u_{j_i}}\right) \\ &= w^u_{j_i}(1-\rho_k) \\ &= \bar{w}_{j_i}(x_{j_i})(1-\rho_k). \end{aligned} \tag{5}$$

According to Lemma 5.1, $l \in \{1, \ldots, m\}$ and $x_{j_i} = 0$ for all $i < l$ and $x_{j_i} = p_j(i)$ exist for all $i \geq l$. Let W'_k and $\ddot{W}_k$ be the total works of rounding solution and the fractional optimal total works in the linear program (2), respectively, at iteration k. For each iteration k, it can be obtained that $x_j = \max_i\{x_{j_i}\} = p_j(l)$ and $\sum_{j=1}^{n} \bar{w}_j(x_j) = \sum_{j=1}^{n}\sum_{i=1}^{m} \bar{w}_{j_i}(x_{j_i}) = \sum_{j=1}^{n} l p_j(l) - \ddot{W}_{k-1} = W'_k - \ddot{W}_{k-1}$. Based on Eq. (5), the upper bound of W'_k is as follows:

$$\begin{aligned} \ddot{W}_k - \ddot{W}_{k-1} &= \sum_{j=1}^{n} \bar{w}_j(x^*_j) \\ &\geq (1-\rho_k)\sum_{j=1}^{n} \bar{w}_j(x_j) \\ &= (1-\rho_k)(W'_k - \ddot{W}_{k-1}) \end{aligned}$$

which yields

$$(1-\rho_k)W'_k + \rho_k \cdot \ddot{W}_{k-1} \leq \ddot{W}_k \tag{6}$$

for $k = 1, \ldots, t$. Based on Eq. (6), since $\ddot{W}_t \leq W^*$, $W' = \min_k\{W'_k\}$, $\ddot{W}_0 = P$, $\rho_1 = \rho$ and $\rho_k = 2\rho$ for $k = 2, \ldots, t$, the following inequality can be obtained

$$\begin{aligned} W' &\leq \frac{W^*}{1-\rho_1\rho_2\cdots\rho_t} - \frac{\rho_1\rho_2\cdots\rho_t}{1-\rho_1\rho_2\cdots\rho_t} P \\ &= \frac{W^*}{1-2^{t-1}\rho^t} - \frac{2^{t-1}\rho^t}{1-2^{t-1}\rho^t} P. \end{aligned}$$

The Lemma is as follows. □

As in Lepére et al. [22], the time interval of the final schedule $[0, C_{\max}]$ is partitioned into three types of time slots. During the first type of time slot, at most $\mu - 1$ processors are busy. During the second type of time slot, at least μ and at most $m - \mu$ processors are busy. During the third type of time slot, at least $m - \mu + 1$ processors are busy. Let T_1, T_2 and T_3 be the corresponding sets of time slots, respectively. Let $|T_i|$ be the overall length of the time slots in set T_i for $i \in \{1, 2, 3\}$. The bound of $|T_1|$ and $|T_2|$ are obtained as follows.

Lemma 5.3. $\rho|T_1| + \min\{\rho, \frac{\mu}{m}\}|T_2| \leq C^*_{\max}$.

Proof. The proof is given in Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2018.2813387. □

The following Lemma shows that the length of any schedule is bounded on P.

Lemma 5.4. $|T_1| + |T_2| + |T_3| \leq P$.

Proof. When all the tasks are scheduled on the same processor, the makespan is $P = \sum_{j=1}^{n} p_j(1)$, which is equal to or larger than any feasible schedule without idle time between tasks. The inequality is proven. □

The upper bound on W is as follows.

Lemma 5.5. $W \leq \frac{W^*}{1-2^{t-1}\rho^t} - \frac{2^{t-1}\rho^t}{1-2^{t-1}\rho^t} P$.

Proof. For each task J_j, the allotment α' allots l'_j processors to task J_j and the allotment α allots $l_j = \min\{l'_j, \mu\}$ processors to task J_j. By Assumption 2, the work of allotment α does not increase. We have $W \leq W'$. By Lemma 5.2, $W \leq W' \leq \frac{W^*}{1-2^{t-1}\rho^t} - \frac{2^{t-1}\rho^t}{1-2^{t-1}\rho^t} P$ can be obtained. □

Let $y_i = |T_i|/C^*_{\max}$ be the normalized overall length of the ith type of time slot for $i = 1, 2, 3$. Also, let $y_p = P/C^*_{\max}$ be the normalized overall length of P. Therefore, we can obtain a min-max nonlinear program to find the optimal approximation ratio of our algorithm.

Lemma 5.6. *The optimal approximation ratio of our algorithm is bounded by the following min-max non-linear program.*

$$\begin{aligned} \min_{\mu,\rho} \quad \max_{y_1,y_2,y_3,y_p} \quad & y_1 + y_2 + y_3 \\ s.t. \quad & y_1 + y_2 + y_3 \leq y_p; \\ & y_1 + \mu y_2 + (m - \mu + 1)y_3 \\ & \leq \frac{m}{1-2^{t-1}\rho^t} - \frac{2^{t-1}\rho^t y_p}{1-2^{t-1}\rho^t}; \\ & \rho y_1 + \min\Big\{\rho, \frac{\mu}{m}\Big\} y_2 \leq 1; \\ & y_1, y_2, y_3, y_p \geq 0; \\ & \rho \in [0, 0.5]; \\ & \mu \in \{1, \ldots, \lfloor (m+1)/2 \rfloor\}. \end{aligned} \tag{7}$$

Proof. The first and third constraints are obtained by normalizing the inequalities in Lemma 5.4 and Lemma 5.3. According to the number of busy processors in each type of time slot, we have $W \geq |T_1| + \mu|T_2| + (m - \mu + 1)|T_3|$. Thus, the second constraint can be obtained by Lemma 5.5. The remaining constraints are obtained from the parameter definitions. According to inequality (4), the approximation ratio is $r = sup\{C_{\max}/OPT\} \leq sup\{C_{\max}/C^*_{\max}\} = sup\{(|T_1| + |T_2| + |T_3|)/C^*_{\max}\} = \max_{y_1,y_2,y_3}\{y_1 + y_2 + y_3\}$, which is our objective function. The goal is to minimize the approximation ratio over all feasible μ and ρ. The Lemma is therefore proven. □

5.1 Approximation Ratio of the Proposed Algorithm

This section analyzes the min-max nonlinear program (7) to obtain an approximation ratio. The following property is used in the approximation ratio analysis.

Proposition 5.1. *Suppose that x_1 is the largest real root to the equation $f(x) = \sum_{i=1}^{n} a_i x^i = 0 (a_n \neq 0)$. If $a_n > 0$, then $f(x) > 0$ for $x \in (x_1, \infty)$. If $a_n < 0$, then $f(x) < 0$ for $x \in (x_1, \infty)$.*

The min-max nonlinear program (7) can be simplified as follows.

Lemma 5.7. *The min-max nonlinear program (7) can be simplified to the following nonlinear program:*

$$\begin{aligned} \min_{\mu,\rho} \max_{y_1,y_2} \ & \frac{y_1(m-\mu)(1-2^{t-1}\rho^t) + y_2(1-2^{t-1}\rho^t)(m-2\mu+1)+m}{(m-\mu)(1-2^{t-1}\rho^t)+1} \\ s.t. \ & \rho y_1 + \min\Big\{\rho, \frac{\mu}{m}\Big\} y_2 \leq 1; \\ & y_1 + y_2(\mu(1-2^{t-1}\rho^t) + 2^{t-1}\rho^t) \leq m; \\ & y_1, y_2 \geq 0; \\ & \rho \in [0, 0.5]; \\ & \mu \in \{1, \ldots, \lfloor (m+1)/2 \rfloor\}. \end{aligned} \tag{8}$$

Proof. The proof is given in Appendix B, available in the online supplemental material. □

According to Lemma 5.7, two cases $\rho < \mu/m$ and $\rho \geq \mu/m$, must be considered. When $t = 1$, our algorithm is the same as Jansen and Zhang [16]. Therefore, as with the analysis in [16], the following bound holds.

Lemma 5.8. *When $t = 1$ and $\rho \leq \mu/m$, our algorithm has an approximation ratio $r \leq m$.*

We then consider the case $\rho \geq \mu/m$. The following min-max nonlinear program must be solved:

$$\begin{aligned} \min_{\mu,\rho} \max_{y_1,y_2} \ & \frac{y_1(m-\mu)(1-2^{t-1}\rho^t) + y_2(1-2^{t-1}\rho^t)(m-2\mu+1)+m}{(m-\mu)(1-2^{t-1}\rho^t)+1} \\ s.t. \ & \text{C1: } \rho y_1 + \frac{\mu}{m} y_2 \leq 1 \\ & \text{C2: } y_1 + y_2(\mu(1-2^{t-1}\rho^t) + 2^{t-1}\rho^t) \leq m \\ & \text{C3: } y_1, y_2, \rho \geq 0 \\ & \rho \in [\frac{\mu}{m}, 0.5] \\ & \mu \in \{1, \ldots, \lfloor (m+1)/2 \rfloor\}. \end{aligned}$$

In a fixed pair ρ and μ, the values of y_1 and y_2 for maximizing the objective value need to be found. The three extreme points $E_1 : (y_1, y_2) = (0, 0), E_2 : (y_1, y_2) = (1/\rho, 0)$, and $E_3 : (y_1, y_2) = (0, m/\mu)$ are obtained by C_1 and C_3. The three extreme points $E_1 : (y_1, y_2) = (0, 0), E_4 : (y_1, y_2) = (m, 0)$, and $E_5 : (y_1, y_2) = (0, m/(\mu(1 - 2^{t-1}\rho^t) + 2^{t-1}\rho^t))$ are obtained by C_2 and C_3.

Since $\mu \geq 1$ and $\rho \geq \mu/m$, we have $\rho \geq \mu/m \geq 1/m$, which yields $1/\rho \leq m$. Therefore, point E_2 is in the polytope defined by E_1, E_4, and E_5. Since $2^{t-1}\rho^t > 0$ and $\mu \geq 1$, $2^{t-1}\rho^t \leq \mu 2^{t-1}\rho^t$, which yields $\mu(1 - 2^{t-1}\rho^t) + 2^{t-1}\rho^t \leq \mu$. We have $m/\mu \leq m/(\mu(1 - 2^{t-1}\rho^t) + 2^{t-1}\rho^t)$. Thus, point E_3 is in the polytope defined by E_1, E_4, and E_5. Constraint C_2 can therefore be removed. The following min-max nonlinear program needs to be considered.

$$\begin{aligned} \min_{\mu,\rho} \max_{y_1,y_2} \ & \frac{y_1(m-\mu)(1-2^{t-1}\rho^t) + y_2(1-2^{t-1}\rho^t)(m-2\mu+1)+m}{(m-\mu)(1-2^{t-1}\rho^t)+1} \\ s.t. \ & \rho y_1 + \frac{\mu}{m} y_2 \leq 1 \\ & y_1, y_2, \rho \geq 0 \\ & \rho \in [\frac{\mu}{m}, 0.5], \\ & \mu \in \{1, \ldots, \lfloor (m+1)/2 \rfloor\}. \end{aligned} \tag{9}$$

Obviously, the maximum value is not at the extreme point E_1. Substituting E_2 and E_3 into the objective function yields

$$A(\mu, \rho) = \frac{1}{\rho} + \frac{m - \frac{1}{\rho}}{(m-\mu)(1-2^{t-1}\rho^t)+1} \tag{10}$$

and

$$B(\mu, \rho) = \frac{\frac{m}{\mu}(1-2^{t-1}\rho^t)(m-2\mu+1)+m}{(m-\mu)(1-2^{t-1}\rho^t)+1}. \tag{11}$$

Function $A(\mu, \rho)$ increases in μ. Function $B(\mu, \rho)$ decreases in μ in the case of $2^{t-1}\rho^t \in (0, 1/2)$. Thus, only one feasible root to the equation $A(\mu, \rho) = B(\mu, \rho)$ can be obtained (as the other exceeds $(m+1)/2$):

$$\mu^* = \frac{m(1+2\rho) - \sqrt{(1+4\rho^2)m^2 - 4m\rho}}{2}. \tag{12}$$

The analysis of $A(\mu, \rho)$ and $B(\mu, \rho)$ is given in Appendix C, available in the online supplemental material.

The following lemma verifies that the constraints $\hat{\mu}^* \leq (m+1)/2$ and $\rho \geq \hat{\mu}^*/m$ are satisfied for $\rho \in [1/3, 1/2]$ and $m \geq 3$.

Lemma 5.9. *For a fixed rounding parameter $\rho \in [1/3, 1/2]$ and $m \geq 3$, $\mu^* \leq (m+1)/2$ and $\rho \geq \mu^*/m$ where $\mu^* = m(1+2\rho)/2 - \sqrt{(1+4\rho^2)m^2 - 4m\rho}/2$.*

Proof. Since μ^* increases in ρ, we only need to show $(2m - \sqrt{2m^2 - 2m})/2 \leq (m+1)/2$, which is equivalent to

$$m - 1 \leq \sqrt{2m^2 - 2m}. \tag{13}$$

Since $m - 1$ and $\sqrt{2m^2 - 2m}$ are positive for $m \geq 1$, the square of both sides yields $m^2 \geq 1$. The roots to $m^2 - 1 = 0$ are ± 1. The inequality (13) holds if $m \geq 1$, according to Proposition 5.1.

Now, we show that $\rho \geq \mu/m = m(1+2\rho)/2m - \sqrt{(1+4\rho^2)m^2 - 4m\rho}/2m$, which is equivalent to

TABLE 2
The Rounding Parameter $\hat{\rho}^*$ for Some t

t	$\hat{\rho}^*(t)$	t	$\hat{\rho}^*(t)$
1	0.4310	100	0.4818
10	0.4314	200	0.4891
20	0.4501	300	0.4920
30	0.4601	400	0.4937
40	0.4665	500	0.4947
50	0.4709	1000	0.4970

$$m \leq \sqrt{(1+4\rho^2)m^2 - 4m\rho}. \quad (14)$$

The square of both sides yields $m \geq 1/\rho$. Since $\rho \in [1/3, 1/2]$, the inequality (14) holds for $m \geq 3$. The Lemma is proven. □

According to Lemma 5.9, the following lemma is obtained.

Lemma 5.10. *For a fixed rounding parameter $\rho \in [1/3, 1/2]$ and $m \geq 3$, the optimal objective value of Eq. (9) is bounded by the optimal objective value for fixed $\mu = \mu^*$ in (12).*

A specific value of ρ can be obtained in Appendix D, available in the online supplemental material. Let $\hat{\rho}^*(t)$ be the optimum value of ρ for t iterations. The value of $\hat{\rho}^*(t)$ for t is as shown in Table 2.

Let $\hat{\mu}^*(t)$ be the optimum value of μ for t iterations. Replacing $\hat{\rho}^*(t)$ into Eq. (12) yields

$$\hat{\mu}^*(t) = \frac{m(1+2\hat{\rho}^*(t)) - \sqrt{(1+4\hat{\rho}^*(t)^2)m^2 - 4m\hat{\rho}^*(t)}}{2}. \quad (15)$$

The following bound of approximation ratio holds.

Lemma 5.11. *In the case $m \geq 3$, our algorithm has an approximation ratio r bounded by*

$$r \leq \frac{1}{\hat{\rho}^*(t)} + \frac{2m - \frac{2}{\hat{\rho}^*(t)}}{(m - 2m\hat{\rho}^*(t) + \sqrt{\Delta})(1-\hat{\rho}_s^*(t)) + 2\hat{\rho}_s^*(t)}$$

where $\Delta = (1+4\hat{\rho}^*(t)^2)m^2 - 4m\hat{\rho}^*(t)$ and $\hat{\rho}_s^*(t) = 2^{t-1}\hat{\rho}^*(t)^t$.

Proof. Since $\hat{\mu}^*$ is a fractional number, two integer values $\lceil \hat{\mu}^* \rceil$ and $\lfloor \hat{\mu}^* \rfloor$ need to be considered. Since $A(\rho, \mu)$ increases in μ and $B(\rho, \mu)$ decreases in μ, the approximation ratio is $r \leq \min\{A(\hat{\rho}^*(t), \lceil \hat{\mu}^*(t) \rceil), B(\hat{\rho}^*(t), \lfloor \hat{\mu}^*(t) \rfloor)\} \leq A(\hat{\rho}^*(t), \hat{\mu}^*(t)+1)$. Therefore, substituting $\hat{\mu}^*(t)$ from Eq. (15) in $A(\hat{\rho}^*(t), \hat{\mu}^*(t)+1)$ gives $r \leq \frac{1}{\hat{\rho}^*(t)} + \frac{2m - \frac{2}{\hat{\rho}^*(t)}}{(m-2m\hat{\rho}^*(t)+\sqrt{\Delta})(1-\hat{\rho}_s^*(t))+2\hat{\rho}_s^*(t)}$. □

Corollary 5.12. *For all $m \in \mathbb{N}$ and $m \geq 2$, the approximation ratio*

$$r \leq \begin{cases} 2, & \text{if } m = 2; \\ 3, & \text{if } m = 3, 4; \\ \frac{1}{\hat{\rho}^*(t)} + \frac{2}{(1-2\hat{\rho}^*(t)+\sqrt{1+4\hat{\rho}^*(t)^2})(1-\hat{\rho}_s^*(t))}, & \text{otherwise} \end{cases}$$

where $\hat{\rho}_s^*(t) = 2^{t-1}\hat{\rho}^*(t)^t$.

Proof. The proof is given in Appendix B, available in the online supplemental material. □

Corollary 5.13. *Given $0 < \epsilon < 0.1$ and $\rho = 1/2 - \epsilon/4$, the approximation ratio of our algorithm is tends to $2+\sqrt{2}$ if $t > \frac{\ln M}{\ln 2\rho}$ where*

$$M = \frac{2 \cdot \left(2 + \sqrt{2} - \frac{1}{\rho} - \frac{2}{1-2\rho+\sqrt{1+4\rho^2}} + \epsilon\right)}{2 + \sqrt{2} - \frac{1}{\rho} + \epsilon}.$$

TABLE 3
The Approximation Ratios of the Proposed Algorithm for $t = 100$

m	μ	ρ	r	m	μ	ρ	r
2	1	0.5000	2.0000	14	5	0.4830	2.8177
3	1	0.3333	3.0000	15	5	0.4829	3.0170
4	2	0.5000	3.0000	16	5	0.4828	3.2164
5	2	0.4853	2.5161	17	6	0.4828	2.8512
6	3	0.4846	2.0224	18	6	0.4827	3.0173
7	3	0.4842	2.3524	19	6	0.4827	3.1835
8	3	0.4839	2.6836	20	7	0.4826	2.8752
9	3	0.4836	3.0156	21	7	0.4826	3.0175
10	4	0.4834	2.5187	22	7	0.4825	3.1599
11	4	0.4833	2.7674	23	7	0.4825	3.3024
12	4	0.4832	3.0165	24	8	0.4825	3.0177
13	5	0.4830	2.6186	25	8	0.4825	3.1423

Proof. Let $\rho_s = 2^{t-1}\rho^t$. According to Corollary 5.12, we must show: given $0 < \epsilon < 0.1$, $|2+\sqrt{2} - \frac{1}{\rho} - \frac{2}{(1-2\rho+\sqrt{1+4\rho^2})(1-\rho_s)}| < \epsilon$ for $t \gg 1$. Since $\frac{1}{\rho} + \frac{2}{(1-2\rho+\sqrt{1+4\rho^2})(1-\rho_s)} \geq 2+\sqrt{2}$, we only need to show $\frac{1}{\rho} + \frac{2}{(1-2\rho+\sqrt{1+4\rho^2})(1-\rho_s)} - 2 - \sqrt{2} < \epsilon$ which is equivalent to

$$M = \frac{2 \cdot \left(2 + \sqrt{2} - \frac{1}{\rho} - \frac{2}{1-2\rho+\sqrt{1+4\rho^2}} + \epsilon\right)}{2 + \sqrt{2} - \frac{1}{\rho} + \epsilon} > (2\rho)^t.$$

Since $2\rho < 1$, the inequality holds if $t > \frac{\ln M}{\ln 2\rho}$. To ensure $\ln M$ is meaningful, we need to show that M is positive. Since $2+\sqrt{2} - \frac{1}{\rho} + \epsilon \geq 0$ for $\rho = 1/2 - \epsilon/4$ and $0 < \epsilon < 0.1$, we only need to show $2+\sqrt{2} - \frac{1}{\rho} - \frac{2}{1-2\rho+\sqrt{1+4\rho^2}} + \epsilon \geq 0$ which is equivalent to show $2 - (1-2\rho)(2+\sqrt{2} - \frac{1}{\rho} + \epsilon) \leq (2+\sqrt{2} - \frac{1}{\rho} + \epsilon)\sqrt{1+4\rho^2}$. Since the both sides of above inequality is positive, we can take the square of both sides of the inequality which yield $(2+\sqrt{2} - \frac{1}{\rho} + \epsilon)^2(1+4\rho^2) - (2 - (1-2\rho)(2+\sqrt{2} - \frac{1}{\rho} + \epsilon))^2 \geq 0$. After simplification the inequality becomes: $E = -\epsilon^3 - 2.8284 \cdot \epsilon^2 + 0.8284 \cdot \epsilon > 0$. Solving $E = 0$, the roots are -3.0960, 0 and 0.2676. The equality M is positive in $\epsilon \in (0, 0.2676)$. Thus, the corollary is proven. □

According to Corollary 5.13, when $\rho = 1/2 - \epsilon/4$, $\epsilon \to 0$ and $t \to \infty$, our algorithm is bounded by $2+\sqrt{2} \approx 3.4143$. For example, when $t = 1000$ and $\hat{\rho}^*(t) = 0.4970$, the approximation ratio is 3.4262, which is very close to 3.4143.

6 Results and Discussion

This section compares the approximation ratio of the proposed algorithm to that of the algorithm in Jansen and Zhang [16]. When $m \to \infty$, Table 2 shows some rounding parameter. In fact, the approximation ratio r can be improved by choosing a rounding parameter of ρ for a fixed m. The rounding parameter is the root of $\sum_{i=0}^{3} c_i m^i$, where c_i is defined in Appendix D, available in the online supplemental material.

TABLE 4
The Approximation Ratios of the Algorithm in Jansen and Zhang [16]

m	μ	ρ	r	m	μ	ρ	r
2	1	0.500	2.0000	14	4	0.430	4.1739
3	1	0.333	3.0000	15	5	0.430	4.2173
4	2	0.500	3.0000	16	5	0.430	4.2065
5	2	0.430	3.3125	17	5	0.430	4.1973
6	2	0.430	3.4458	18	6	0.430	4.3249
7	3	0.430	3.7507	19	6	0.430	4.3083
8	3	0.430	3.7995	20	6	0.430	4.2938
9	3	0.430	3.8356	21	6	0.430	4.2880
10	3	0.430	3.9078	22	7	0.430	4.3857
11	4	0.430	4.0639	23	7	0.430	4.3685
12	4	0.430	4.0656	24	7	0.430	4.3531
13	4	0.430	4.0669	25	7	0.430	4.3897

Since $\hat{\mu}^*$ is a fractional number, two integer values $\lceil \hat{\mu}^* \rceil$ and $\lfloor \hat{\mu}^* \rfloor$ need to be considered. The approximation ratio obtained using the proposed algorithm is bounded by $\min\{A(\hat{\rho}^*(t), \lceil \hat{\mu}^*(t) \rceil), B(\hat{\rho}^*(t), \lfloor \hat{\mu}^*(t) \rfloor)\}$. Table 3 lists the values of the bounds on the proposed algorithm under Assumptions 1 and 2 for $t = 100$. Table 4 lists the approximation ratios obtained using Jansen and Zhang's [16] algorithm under the same assumptions. The proposed algorithm obviously improves on Jansen and Zhang's [16] results..

Table 5 lists the values of the bounds on the proposed algorithm for $m = 25$. Note that, when $t = 1$, our algorithm is the same as Jansen and Zhang [16]. However, the first line of Table 5 is not consistent with the last line of Table 4 because the rounding parameter in Table 4 is obtained by setting $m \to \infty$, and the rounding parameter in Table 5 is obtained by setting $m = 25$. The rounding parameter ρ increases in $t \geq 3$, and the approximation ratio decreases in t. As the number of iterations increases, the rounding parameter is tended to $0.4\bar{9}$, and the approximation ratio is tended to 3.1267. According to Corollary 5.12, Table 6 lists the values of the bounds on the proposed algorithm for any m. As the number of iterations increases, the approximation ratio r is improved from 4.7306 to 3.4143. In the iteration 1 to 50, the proposed algorithm significantly improves the approximation ratio r. For a large number of iterations (> 100), the approximation ratio of the proposed algorithm is tended to $2 + \sqrt{2} \approx 3.4143$.

TABLE 5
The Approximation Ratios of the Proposed Algorithms for $m = 25$

t	μ	ρ	r	t	μ	ρ	r
1	8	0.4512	4.0807	60	8	0.4752	3.1542
2	8	0.4215	3.7747	70	8	0.4776	3.1499
3	7	0.4179	4.0267	80	8	0.4795	3.1467
4	7	0.4194	3.9360	90	8	0.4811	3.1443
5	8	0.4221	3.4517	100	8	0.4825	3.1423
6	8	0.4251	3.4044	200	8	0.4894	3.1335
7	8	0.4281	3.3687	300	8	0.4923	3.1307
8	8	0.4309	3.3409	400	8	0.4938	3.1292
9	8	0.4335	3.3187	500	8	0.4948	3.1284
10	8	0.4360	3.3005	600	8	0.4955	3.1278
20	8	0.4527	3.2142	700	8	0.4961	3.1274
30	8	0.4620	3.1843	800	8	0.4965	3.1271
40	8	0.4679	3.1692	900	8	0.4968	3.1269
50	8	0.4720	3.1602	1000	8	0.4971	3.1267

TABLE 6
The Approximation Ratios of the Proposed Algorithms for any m

t	ρ	r	t	ρ	r
1	0.4310	4.7306	60	0.4742	3.5373
2	0.4083	4.4841	70	0.4767	3.5233
3	0.4077	4.3131	80	0.4788	3.5124
4	0.4108	4.1931	90	0.4804	3.5036
5	0.4147	4.1043	100	0.4818	3.4964
6	0.4185	4.0358	200	0.4891	3.4610
7	0.4222	3.9811	300	0.4920	3.4476
8	0.4255	3.9362	400	0.4937	3.4405
9	0.4286	3.8987	500	0.4947	3.4360
10	0.4314	3.8668	600	0.4954	3.4328
20	0.4501	3.6955	700	0.4960	3.4305
30	0.4601	3.6233	800	0.4964	3.4288
40	0.4665	3.5826	900	0.4967	3.4274
50	0.4709	3.5561	1000	0.4970	3.4262

7 Concluding Remarks

This article developed a polynomial-time approximation algorithm to solve the problem of scheduling malleable tasks with precedence constraints and presented an iterative method for improving the performance ratio of scheduling malleable tasks. The concept of the proposed algorithm is to reduce the gap between the fractional optimal solution and the integral optimal solution. The proposed algorithm achieves an approximation ratio of 4.4841 after 2 iterations, thus improving on the best-known factor of 4.7306 from Jansen and Zhang [16]. For a large number of iterations (> 100), the approximation ratio of the proposed algorithm is tended to $2 + \sqrt{2} \approx 3.4143$.

Acknowledgments

The work was supported by the Ministry of Science and Technology, Taiwan, R.O.C. under Grant no. MOST 106-2221-E-006-006.

References

[1] M. S. Barketau, M. Y. Kovalyov, J. Węglarz, and M. Machowiak, "Scheduling arbitrary number of malleable tasks on multiprocessor systems," *Bulletin Polish Acad. Sci. Tech. Sci.*, vol. 62, no. 2, 2014, Art. no. 255.

[2] E. Blayo, L. Debreu, G. Mounié, and D. Trystram, "Dynamic load balancing for ocean circulation model with adaptive meshing," in *Proc. 5th Eur. Conf. Parallel Comput.*, 1999, pp. 303–312.

[3] P. Bodik, I. Menache, J. S. Naor, and J. Yaniv, "Brief announcement: Deadline-aware scheduling of big-data processing jobs," in *Proc. 26th ACM Symp. Parallelism Algorithms Archit.*, 2014, pp. 211–213.

[4] E. Caron and M. D. de Assunção, "Multi-criteria malleable task management for hybrid-cloud platforms," in *Proc. 2nd Int. Conf. Cloud Comput. Technol. Appl.*, May 2016, pp. 326–333.

[5] R. Chaiken, B. Jenkins, P.-A. Larson, B. Ramsey, D. Shakib, S. Weaver, and J. Zhou, "Scope: Easy and efficient parallel processing of massive data sets," *Proc. VLDB Endow.*, vol. 1, no. 2, pp. 1265–1276, Aug. 2008.

[6] C.-Y. Chen and C.-P. Chu, "A 3.42-approximation algorithm for scheduling malleable tasks under precedence constraints," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 8, pp. 1479–1488, Aug. 2013.

[7] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[8] T. Decker, T. Lücking, and B. Monien, "A 5/4-approximation algorithm for scheduling identical malleable tasks," *Theor. Comput. Sci.*, vol. 361, no. 2, pp. 226–240, Sep. 2006.
[9] J. Du and J. Y.-T. Leung, "Complexity of scheduling parallel task systems," *SIAM J. Discrete Math.*, vol. 2, no. 4, pp. 473–487, 1989.
[10] M. R. Garey and R. L. Graham, "Bounds for multiprocessor scheduling with resource constraints," *SIAM J. Comput.*, vol. 4, no. 2, pp. 187–200, 1975.
[11] R. L. Graham, "Bounds for certain multiprocessing anomalies," *Bell Syst. Tech. J.*, vol. 45, no. 9, pp. 1563–1581, 1966.
[12] K. Jansen, "Scheduling malleable parallel tasks: An asymptotic fully polynomial-time approximation scheme," *Algorithmica*, vol. 39, no. 1, pp. 59–81, 2004.
[13] K. Jansen, "A$(3/2+\epsilon)$ approximation algorithm for scheduling moldable and non-moldable parallel tasks," in *Proc. 24th Annu. ACM Symp. Parallelism Algorithms Architectures*, 2012, pp. 224–235.
[14] K. Jansen and L. Porkolab, "Linear-time approximation schemes for scheduling malleable parallel tasks," *Algorithmica*, vol. 32, no. 3, pp. 507–520, 2002.
[15] K. Jansen and R. Thöle, "Approximation algorithms for scheduling parallel jobs," *SIAM J. Comput.*, vol. 39, no. 8, pp. 3571–3615, 2010.
[16] K. Jansen and H. Zhang, "An approximation algorithm for scheduling malleable tasks under general precedence constraints," *ACM Trans. Algorithms*, vol. 2, no. 3, pp. 416–434, 2006.
[17] K. Jansen and H. Zhang, "Scheduling malleable tasks with precedence constraints," *J. Comput. Syst. Sci.*, vol. 78, pp. 245–259, 2011.
[18] Y.-K. Kwok and I. Ahmad, "Dynamic critical-path scheduling: An effective technique for allocating task graphs to multiprocessors," *IEEE Trans. Parallel Distrib. Syst.*, vol. 7, no. 5, pp. 506–521, May 1996.
[19] Y.-K. Kwok and I. Ahmad, "Static scheduling algorithms for allocating directed task graphs to multiprocessors," *ACM Comput. Surv.*, vol. 31, no. 4, pp. 406–471, Dec. 1999.
[20] J. K. Lenstra and A. H. G. Rinnooy Kan, "Complexity of Scheduling under Precedence Constraints," *Oper. Res.*, vol. 26, no. 1, pp. 22–35, 1978.
[21] R. Lepére, G. Mounié, and D. Trystram, "An approximation algorithm for scheduling trees of malleable tasks," *Eur. J. Oper. Res.*, vol. 142, no. 2, pp. 242–249, 2002.
[22] R. Lepére, D. Trystram, and G. J. Woeginger, "Approximation algorithms for scheduling malleable tasks under precedence constraints," *Int. J. Found. Comput. Sci.*, vol. 13, no. 4, pp. 613–627, 2002.
[23] W. Ludwig and P. Tiwari, "Scheduling malleable and nonmalleable parallel tasks," in *Proc. 5th Annu. ACM-SIAM Symp. Discrete Algorithms*, 1994, pp. 167–176.
[24] G. Mounié, C. Rapine, and D. Trystram, "A $\frac{3}{2}$-approximation algorithm for scheduling independent monotonic malleable tasks," *SIAM J. Comput.*, vol. 37, no. 2, pp. 401–412, 2007.
[25] G. Mounié, C. Rapine, and D. Trystram, "Efficient approximation algorithms for scheduling malleable tasks," in *Proc 11th Annu ACM Symp Parallel Algorithms Architectures*, 1999, pp. 23–32.
[26] M. Skutella, "Approximation algorithms for the discrete time-cost tradeoff problem," *Math. Oper. Res.*, vol. 23, no. 4, pp. 909–929, 1998.
[27] X. Tang, K. Li, and D. Padua, "Communication contention in apn list scheduling algorithm," *Sci. China Series F: Inf. Sci.*, vol. 52, no. 1, pp. 59–69, Jan. 2009.
[28] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans Parallel Distrib. Syst*, vol. 13, no. 3, pp. 260–274, Mar. 2002.
[29] J. Turek, J. L. Wolf, and P. S. Yu, "Approximate algorithms scheduling parallelizable tasks," in *Proc. 4th Annu. ACM Symp. Parallel Algorithms Architectures*, 1992, pp. 323–332.
[30] X. Wu and P. Loiseau, "Algorithms for scheduling deadline-sensitive malleable tasks," in *Proc. 53rd Annu. Allerton Conf. Commun. Control Comput.*, Sep. 2015, pp. 530–537.

Chi-Yeh Chen received the BS degree in communication engineering from Da-Yeh University, Changhua, Taiwan, ROC, in 2001, the MS degree in computer science and information and engineering from National Cheng Kung University, Tainan, Taiwan, ROC, in 2005, and the PhD degree in computer science and information and engineering from National Cheng Kung University, Tainan, Taiwan, ROC, in 2012. He is currently an assistant researcher with the Department of Computer Science and Information Engineering, National Cheng Kung University, Taiwan, ROC. His research interests include scheduling problems, approximation algorithms, parallel algorithm, and load distribution.